26장. 코틀린으로 자바스크립트 웹 애플리케이션 개발

26.1. 프런트엔드 웹 애플리케이션

26.1.1. DOM Node 핸들링

DOM 객체 획득

- `external val document: Document`

```
val node = document.getElementById("result_1_1")
node?.innerHTML="<a href='https://kotlinlang.org'>kotlin</a>"
```

- getElementById : 태그의 id 속성 값을 조건으로 객체 획득
- getElementsByClassName : 태그의 css class명을 조건으로 객체 획득
- getElementsByTagName : 태그의 이름을 조건으로 객체 획득

26.1. 프런트엔드 웹 애플리케이션

Event 등록

- fun addEventListener(type: String, callback: (Event) -> Unit, options: dynamic = definedExternally)

```
<button id="btn_1_1" onclick="Ch26_main.twenty_six_one_one.sayHello()">sayHello</button>
```

```
external fun alert(msg: String): Unit

val sayHello = fun (event: Event){
    alert("Hello Kotlin")
}
fun selectDOMNode(){
    val button= document.getElementById("btn_1_2")
    button?.addEventListener("click", sayHello)
}
```

26.1. 프런트엔드 웹 애플리케이션

DOM Node Create

```
val result = document.getElementById("result_1_2")
var html="<ul>"
array.forEach {
    html += "<li>$it</li>"
}
html += "</ul>"
result?.innerHTML=html
```

```
val ul = document.createElement("ul")
array.forEach {
    val li= document.createElement("li")
    li.textContent=it
    ul.appendChild(li)
}
result?.appendChild(ul)
```

26.1. 프런트엔드 웹 애플리케이션

26.1.2. Ajax 프로그래밍

```
external class XMLHttpRequest

fun ajax() {
    var xhr :dynamic= XMLHttpRequest();
    xhr.open("GET", "some.json", true);
    xhr.onreadystatechange=fun(){
        if (xhr.status == 200) {
            println(xhr.responseText)
        }else {
            println("error")
        }
    }
    xhr.send();
}
```

26.2. DSL을 위한 kotlinx-html-js 라이브러리 이용

26.2.1. kotlinx-html-js 라이브러리 소개

- kotlinx-html-js 를 통해 코틀린에서도 조금더 쉽게 HTML 태그를 생성
- kotlinx-html-js 라이브러리는 kotlinx.html 라이브러리의 한 종류
- kotlinx.html 라이브러리는 kotlinx-html-js와 kotlinx-html-jvm 으로 구분

```
repositories {
    mavenCentral()
    maven {
        url  "http://dl.bintray.com/kotlin/kotlinx.html/"
    }
}

dependencies {
    compile 'org.jetbrains.kotlinx:kotlinx-html-js:0.6.3'
    //.....................................
}
```

26.2. DSL을 위한 kotlinx-html-js 라이브러리 이용

26.2.2. kotlinx-html-js 을 이용한 DOM Node 동적 생성

```
fun createDOM(){
    val root= document.getElementById("result_2_1")

    val div= document.createElement("div")
    div.addClass("panel")

    val p = document.createElement("p")
    p.addClass("pClass")
    p.textContent="Hello"

    val a = document.createElement("a")
    a.setAttribute("href","http://kotlinlang.org")
    a.textContent="kotlin"

    val button = document.createElement("button")
    button.textContent="click me!"
    button.addEventListener("click", { println("button click...")})

    div.appendChild(p)
    div.appendChild(a)
    div.appendChild(button)

    root?.appendChild(div)
}
```

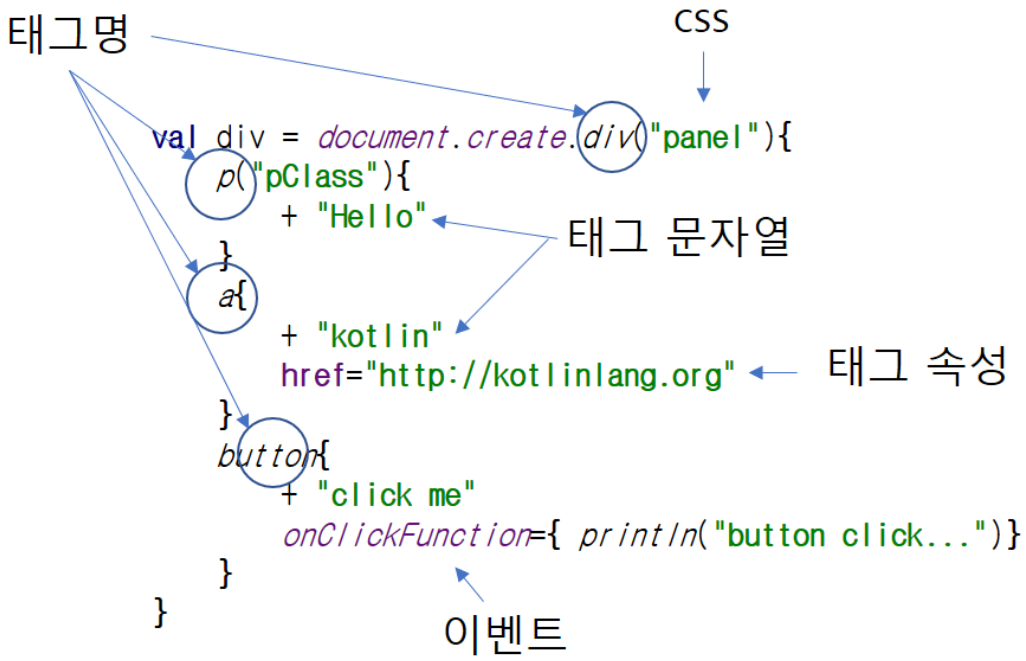
26.2. DSL을 위한 kotlinx-html-js 라이브러리 이용

```
fun useDSL(){
    val root= document.getElementById("result_2_1")

    val div = document.create.div("panel"){
        p("pClass"){
            + "Hello"
        }
        a{
            + "kotlin"
            href="http://kotlinlang.org"
        }
        button{
            + "click me"
            onClickFunction={ println("button click...")}
        }
    }

    root?.appendChild(div)
}
```

26.2. DSL을 위한 kotlinx-html-js 라이브러리 이용




```
<script type="text/javascript" src="../out/production/classes/lib/kotlin.js"></script>
<script type="text/javascript" src="../out/production/classes/lib/kotlinx-html-js.js"></script>
<script type="text/javascript" src="../out/production/classes/Ch26_main.js"></script>
```

26.2. DSL을 위한 kotlinx-html-js 라이브러리 이용

DOM Tree 표현

- DOM 함수의 { } 안에 하위 태그 내용을 명시

```
val myDiv = document.create.div {
    p {
        +"p one"
        div {
            + "sub div"
        }
    }
    p {
        +"p two"
    }
}
```

CSS Class 추가

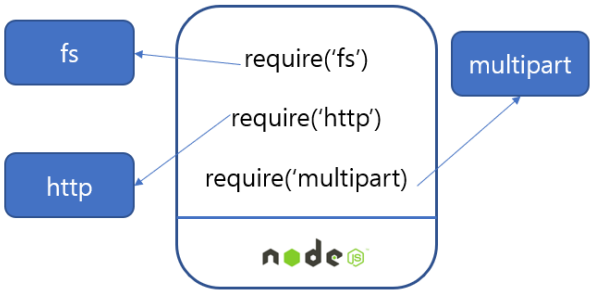
- DOM의 ( ) 안에 매개변수로 지정

```
val myDiv = document.create.div {
    p(classes = "container left tree")
    p {
        classes=setOf("container", "left", "tree")
        classes += "siteHeader"
    }
}
```

26.3. NodeJS로 백엔드 웹 애플리케이션 개발

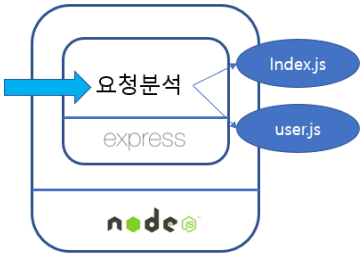
26.3.1. NodeJS와 Express 모듈의 이해

- NodeJS는 Back-End Web Application을 위한 자바스크립트 플랫폼
- NodeJS에 다양한 모듈을 추가 설치하여 자바스크립트 프로그램에서 필요한 모듈을 require() 함수를 이용하여 로딩해서 사용



Back-End Web Application

- Express란 NodeJS를 위한 MVC 프레임워크로 클라이언트로부터 HTTP 요청이 들어오면 요청을 분석



Back-End Web Application

26.3. NodeJS로 백엔드 웹 애플리케이션 개발

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('listening on port 3000!');
});
```

```
external fun require(module:String):dynamic

fun main(args: Array<String>) {

    val express = require("express")
    val app = express()

    app.get("/", { req, res ->
        res.type("text/plain")
        res.send("HelloWorld with Kotlin")
    })

    app.listen(3000, {
        println("Listening on port 3000")
    })
}
```